ELMo vs GPT vs BERT

Jun Gao

Tencent AI Lab

October 18, 2018

Overview

Background

ELMo

GPT

BERT

Background

Language model pre-training has shown to be effective for improving many natural language processing.

- sentence-level: natural language inference and paraphrasing
- token-level: named entity recognition and SQuAD question answering

There are two existing strategies for applying pre-trained language representations to dwonstream tasks.

- feature-based: ELMo
- fine-tuning: GPT
- feature-based & fine-tuning: BERT

Background

Language models are typically left-to-right.(GPT)

- too$\rightarrow$young$\rightarrow$too$\rightarrow$simple$\rightarrow$sometimes$\rightarrow$[naive]

If each word can only see context to its left, clearly a lot is missing. So one trick that people have done is to also train a right-to-left model. (context2vec)

- too$\rightleftharpoons$young$\rightleftharpoons$too$\rightleftharpoons$[simple]$\rightleftharpoons$sometimes$\rightleftharpoons$naive(biLSTM)

Words can indirectly "see themselves", and the predictions become trivial.

- ELMo: too$\rightleftharpoons$young$\rightleftharpoons$too [simple] sometimes$\rightleftharpoons$naive(biLM)
- BERT: too$\rightleftharpoons$[mask1]$\rightleftharpoons$too$\rightleftharpoons$[mask2]$\rightleftharpoons$sometimes$\rightleftharpoons$naive

Outline

Background

ELMo

GPT

BERT

ELMo: Introduction

Previous work has shown that different layers of deep biRNNs encode different types of information.

- Higher-level LSTM states capture context-dependent aspects of word meaning.
- Lower-level states model aspects of sytnax.

ELMo: Bidirectional language model

Given a sequence of N tokens, $(t_1, t_2, ..., t_N)$, a foward language model computes the probability of the sequence by modeling the probability of token t_k given the history $(t_1, ..., t_{k-1})$

$$p(t_1, t_2, ...t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, ..., t_{k-1})$$

A backward LM is similar to a forward LM

$$p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} (t_k | t_{k+1}, t_{k+2}, ..., t_N)$$

ELMo: Embeddingsfrom Language Models

For each token t_k, a L-layer biLM computes a set of $2L+1$ representations

$$\begin{aligned} R_k &= \{x_k^{LM}, \overrightarrow{h}_{k,j}^{LM}, \overleftarrow{h}_{k,j}^{LM} | j = 1, ..., L\} \\ &= \{h_{k,j}^{LM} | j = 0, ..., L\} \end{aligned}$$

For a specific down-stream task, ELMo would learn a weight to combine these representations

$$ELMo_k^{task} = E(R_k|\Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}^{LM}$$

ELMo: Using biLMs for supervised NLP tasks

Add ELMo at the input of RNN. Forsome tasks (SNLI, SQuAD), including ELMo at the output brings further improvements

Keypoint:

- freezethe weight of the biLM
- Regularization is necessary: $\lambda ||w||_2^2$

$$ELMo_k^{task} = E(R_k|\Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}^{LM}$$

ELMo: Conclusion

- Language Modeling is effective in constructing contextualized representation (could be helpful for a variety of tasks);
- Outputs of all Layers are useful;

Outline

GPT: Introduction

- Language Modeling is effective in constructing contextualized representation (could be helpful for a variety of tasks)
- Outputs of all Layers are useful

GPT: Framework

- Unsupervised pre-training

$$L_1(\mu) = \sum_i logP(u_i|u_{i-k}, ..., u_{i-1}; \Theta)$$

$$h_0 = UW_e + W_p$$

$$h_l = transformer_block(h_{l-1}) \forall i \in [1, n]$$

$$P(u) = softmax(h_n W_e^T)$$

- Supervised fine-tuning

$$P(y|x^1, ..., x^m) = softmax(h_l^m W_y)$$

$$L_2(c) = \sum_{(x,y)} logP(y|x^1, ..., x^m)$$

$$L_3(c) = L_2(C) + \lambda * L_1(c)$$

Overall, the only extra parameters we require during fine-tuning are W_y, and embeddings for delimiter tokens.

GPT: Task-specific input transformations

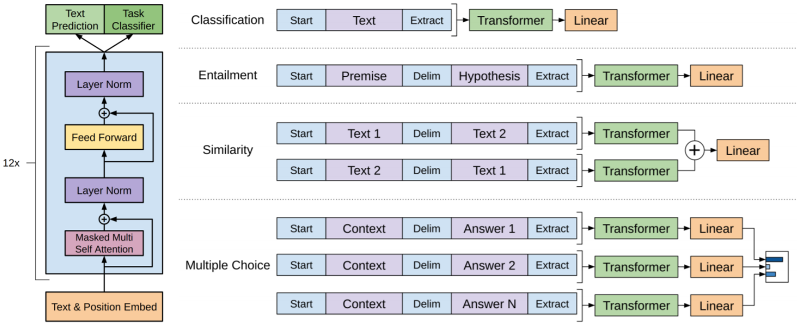


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

Outline

BERT: Introduction

- GPT is unidirectional
- ELMo requires training a separate model

Intuitively, it would be much better if we could train a single model that was deeply bidirectional.

BERT-GPT-ELMo

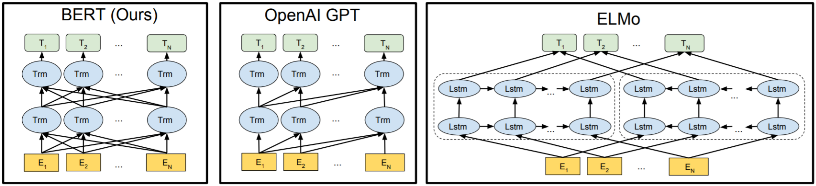


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

BERT : Input representation

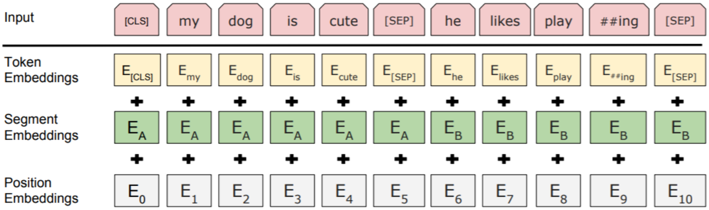


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

BERT: Pre-training Task 1 Masked LM

Mask 15% of all tokens in each sequence at random.

- Input: the man [MASK1] to [MASK2] store
- Label: [MASK1] = went; [MASK2] = store

Rather than always replacing the chosen words with [MASK], the data generator will do the following:

- 80% of the time: Replace the word with the[MASK] token, e.g., my dog is hairy →my dog is [MASK]
- 10% of the time: Replace the word with arandom word, e.g., my dog is hairy → my dog is apple
- 10% of the time: Keep the word unchanged,e.g., my dog is hairy → my dog is hairy. The purpose of this is to bias the representation towards the actual observed word.

BERT: Pre-training Task 2 Next Sentence Prediction

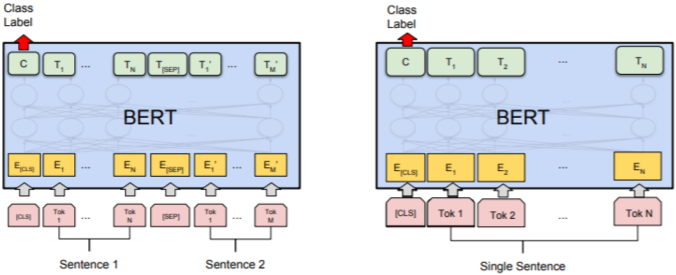
The other thing that's missing from an LM is that it doesn't understand relationships between sentences, which is important for many NLP tasks.
Pre-train a binarized next sentence prediction task.

- Input: the man went to the store [SEP] he bought a gallon of milk
- Label: IsNext
- Input: the man went to the store [SEP] penguins are flightless birds
- Label: NotNext

BERT: Fine-tuning Procedure

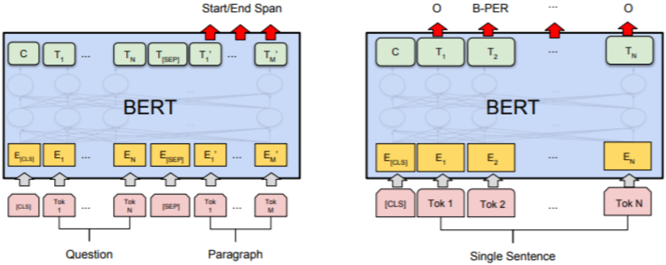
- Take the final hidden state for the first token [cls] in the input as the representation of the input sequence.
- The The only new parametersadded during fine-tuning are for a classification layer
- All of the parameters are fine-tuned jointly to maximize the log-probability of the correct label.

BERT: Fine-tuning Procedure



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA



(c) Question Answering Tasks:

(d) Single Sentence Tagging Tasks:

BERT: Feature-based Approach with BERT

Layers	Dev F1
Finetune All	96.4
First Layer (Embeddings)	91.0
Second-to-Last Hidden	95.6
Last Hidden	94.9
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1
Sum All 12 Layers	95.5

Table 7: Ablation using BERT with a feature-based approach on CoNLL-2003 NER. The activations from the specified layers are combined and fed into a two-layer BiLSTM, without backpropagation to BERT.